Desenvolvimento de blocos de codificação Run-Length Limited usando plataforma de Software-Defined Radio para comunicação por luz visível segundo padrão IEEE 802.15.7

Pedro Henrique Moura da Rosa
GEDRE – *Inteligência em Iluminação*
Universidade Federal de Santa Maria
Santa Maria, Brasil
pedro.moura@acad.ufsm.br

Vitalio Alfonso Reguera
GEDRE – Inteligência em Iluminação
Universidade Federal de Santa Maria
Santa Maria, Brasil
vitalio.reguera@ufsm.br

Carlos Henrique Barriquello
GEDRE – *Inteligência em Iluminação*
Universidade Federal de Santa Maria
Santa Maria, Brasil
barriquello@gedre.ufsm.br

***Resumo* — Software-defined radio (SDR) são sistemas de comunicação, usualmente aplicados a transmissão de sinal via rádio, aonde o processamento de sinal é realizado por um computador pessoal ou um sistema embarcado, ao invés de ser feito por circuitos eletrônicos. Este trabalho tem por objetivo a obtenção de resultados experimentais, quanto a transmissão de dados de um sistema de comunicação por luz visível, a partir da utilização do software GNU Radio para controle do transmissor e análise das amostras obtidas no receptor. Deste modo serão apresentados os blocos utilizados para cada função no processamento de sinal, onde além de contar com os blocos nativos do GNU Radio serão desenvolvidos os blocos Run-Length Limited (RLL), a fim de obter uma amostra do comportamento esperado pelo receptor (Rx) e transmissor (Tx). A documentação para a codificação destes blocos é encontrada na IEEE Standard 802.15.7-2018 Ch. 10 – PHY I Specifications, aonde se tem a rotina de operação do Tx-VLC dividida em blocos de Foward Error Correction (FEC), Interleaving e Puncturing, e por ultimo os RLL.**

Palavras-chave— SDR. LEDs. VLC. Blocos de Programação

I. INTRODUÇÃO

A sociedade contemporânea é marcada pela competitividade e consequente necessidade de informações em tempo real necessitando assim de agentes capazes de fornecerem dados em velocidades mais altas e de forma conveniente. Até então várias soluções foram propostas para a transmissão de dados, sendo divididas pelo tipo do canal de comunicação sendo este com ou sem fio, onde como exemplo pode-se citar a fibra óptica e a comunicação por rádio frequência (RF), respectivamente. Neste trabalho, o foco de estudo se dará na comunicação sem fio, com foco na mobilidade dos usuários, com a tecnologia abordada sendo, mais especificamente, a comunicação por luz visível (VLC).

Casado com a evolução da tecnologia de iluminação baseada em Diodos Emissores de Luz (LEDs) o VLC surge como uma alternativa a RF para ambientes em que sua aplicação seja conveniente e prática. Ao utilizar os LEDs como transmissores (Tx) é possível realizar a modulação da luz gerada devido a capacidade inerente dos LEDs de variar sua intensidade luminosa, alternando entre níveis altos e baixos, numa velocidade alta de modo que não seja perceptível aos olhos humanos. Além disso, o aumento significativo na vida útil da iluminação LED, variando de 25.000 a 50.000 horas, comparada com a fluorescente compacta (10.000 horas) tornou muito mais atrativo a sua utilização. Devido a estes benefícios e a adoção constante pela tecnologia, espera-se que 75% de toda iluminação vai ser baseada em LEDs até o ano de 2030 [1].

Com a dupla função dos LEDs, tanto para iluminação quanto para comunicação, o VLC se mostra um aliado à economia de recursos, possibilitando o *retrofit* da tecnologia e aumentando ainda mais a atratividade do sistema. Com base em sua topologia e codificação é viável aplicar o VLC às mais diversas situações, seja para a transmissão de dados à alta velocidade para download de informações massivas, quanto a baixa velocidade para sinalização. Deste modo é possível caracterizar o VLC quanto a sua aplicabilidade e então usá-lo no melhor cenário, sendo estes *indoor* ou *outdoor*.

Para realizar o envio de dados, o transmissor (Tx) precisa receber as informações necessárias para a construção da mensagem através de um sinal elétrico modulado. Deste modo, variando-se a potência no LED, pode-se obter níveis altos e baixos, os quais serão captados pelo receptor (Rx) e então decodificados num arranjo especifico para a reconstrução do sinal.

Nesse sentido, este trabalho está organizado de uma maneira na qual primeiramente será apresentado como o sinal modulado é gerado a partir da caracterização da camada PHY. Após isto será descrito como o SDR pode ser utilizado em conjunto com o VLC para que no fim possam ser exibidos alguns testes realizados utilizando a topologia apresentada com o *software* GNU Radio realizando o processamento digital para o comando do USRP-2944 (do inglês Universal Software Radio Peripheral) a partir do padrão contido em [2] para a camada PHY.

II. SOFTWARE-DEFINED RADIO

Comumente utilizado na indústria por fornecer agilidade na prototipagem [3] o SDR é uma metodologia de controle de sinal prática que faz digitalmente o tratamento dos dados ao invés de utilizar circuitos eletrônicos. A partir de conversores analógicos digitais e um computador com processador e memória, o SDR recebe e envia informações, permitindo então analisar as amostras de dados através de programação lógica.

Em [4] é utilizada uma versão modificada do USRP2 da Ettus Reasearch [5] devido a relação Custo-Performance do dispositivo, objetivando fazer a reprodução de vídeos utilizando a modo de operação da IEEE 802.15.7 para camada

PHY-I. De forma semelhante em [6] são implementados dois canais de comunicação utilizando OFDM em placas WARP [7] para VLC, demonstrando a constelação gerada e a resposta em frequência.

Neste trabalho, a abordagem será parecida com a dos trabalhos em [4] [6], aonde se utiliza o USRP para o controle do LED e validar a recepção do sinal, assim se consegue checar o funcionamento dos blocos de controle adicionados ao *software* GNU Radio [8] verificando assim a integridade dos arquivos e da operação do sistema.

III. STANDARD IEEE 802.15.7

A IEEE 802.15.7 [2] foi elaborada como um padrão para definir algumas técnicas base para a construção do VLC a curtas distancias. O padrão subdivide a camada PHY em seis tipos, sendo estas a PHY-I, PHY-II, PHY-III, PHY-IV, PHY-V e PHY-V de acordo as características apresentadas na seção III deste trabalho.

A camada PHY-I é constituída por dois tipos de modulações sendo estas a OOK e a VPPM (do inglês *Variable Pulse Position Modulation*) subdividindo-se assim em duas categorias. Ao utilizar códigos corretores de erro em série, como o *Reed-Solomon* (RS) e o *Convolutional Coding* (CC) para cada uma das modulações, a camada PHY-I se separa em 8 modos de operação que variam de acordo com as configurações dos corretores de erro e da modulação selecionada [2].

Além dos fatores citados, a camada PHY-I suporta também dois tipos de códigos RLL sendo estes o Manchester e o 4B6B. Estes códigos são responsáveis por manter o controle DC, recuperar o *clock* e mitigar os efeitos do *flicker* [3].

A Fig.1 mostrada abaixo revela as configurações estabelecidas para a camada PHY-I:

PHY I Mode	Data Rate kb/s	Modul.	RLL code	Optical clock rate	FEC	
					Outer code (RS)	Inner code (CC)
a	11.67	OOK	Manchester	200 kHz	(15,7)	1/4
b	24.44				(15,11)	1/3
c	48.89				(15,11)	2/3
d	73.3				(15,11)	None
e	100				None	None
f	35.56	VPPM	4B6B	400 kHz	(15,2)	None
g	71.11				(15,4)	None
h	124.4				(15,7)	None
i	266.6				None	None

Fig.1. Modos de Operação PHY-I [2]

Na Fig. 2 é exibido o fluxograma para o processamento do sinal e o envio dos dados ao transmissor. Os bits entram primeiramente no codificador RS aonde são rearranjados em uma nova estrutura. Na camada PHY-I o código RS apresenta quatro configurações diferentes, sendo estas RS (15,11), RS (15,7), RS (15,4) e RS (15,2) com o *Galois Field* de (2^4) [9]. O bloco *interleaver* é colocado entre os blocos corretores de erro a fim de aumentar a performance do sistema. Com um tamanho "n" fixo mas profundidade "D" variável em função do tamanho do *frame* enviado, o *interleaver* faz um cruzamento entre os dados do *bitstream* aliviando os *burst errors*. Junto dele é implementado um bloco de *puncturing* para minimizar os erros durante o preenchimento do vetor de bits [2]. Após, o CC gera os símbolos de paridade com a taxa de 1/3 e tamanho máximo K = 7. As taxas de 1/4 e 2/3 são obtidas pela combinação do CC com o *puncturing*. Por fim, tem-se os códigos RLL, sendo estes o Manchester ou o 4B6B, onde para a modulação OOK se usa somente o Manchester e para a VPPM somente o 4B6B. Enquanto o código Manchester duplica o *bitstream* ao expandir cada bit para um símbolo 2-bit, o código 4B6B aumenta o tamanho do vetor em 1,5 vezes, ao expandir o conjunto de 4-bits para 6-bits [2].

Assim, o *bitstream* finalmente chega no bloco modulador seja este OOK ou VPPM. Para a modulação OOK os bits são representados através do nível 'alto' e 'baixo', 'ligado' e 'desligado', '1' e '0'. Em contrapartida a modulação VPPM define os dados através da borda de 'subida' ou de 'descida' do seu sinal, isto é, na transição entre um nível e outro [3]. Após isto, o sinal vai para o canal onde sofre alterações devido a ruídos do meio e é captado pelo receptor, responsável de enviar os dados que chegam ao SDR para que seja feita a demodulação do sinal e o tratamento inverso para cada um dos blocos utilizados na modulação.

A partir do esquemático apresentado é possível processar o sinal VLC via SDR.

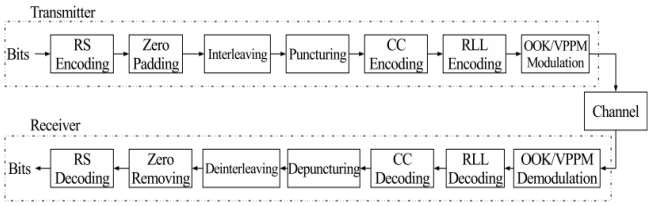

Fig.2. Fluxograma do processamento de sinal da camada PHY-I [10]

IV. EXPERIMENTAÇÃO COM GNU RADIO SDR APLICADO AO VLC

Para este trabalho o *software* adotado para trabalhar foi o GNU Radio [8]. Por contar com uma arquitetura baseada em blocos de controle, o GNU Radio é um *software open source* que permite ao usuário interagir com uma gama de equipamentos analógicos e passar comandos para os mesmos. Através de *drivers* específicos, o GNU Radio realiza a conexão com micro-computadores, USRPs, equipamentos de áudio e vídeo, entre outros.

Além de contar com uma biblioteca nativa contendo blocos interessantes para a utilização no VLC, o GNU Radio permite a criação de novos blocos através de programação em Python ou C++ com a ferramenta 'gr_modtool'. Deste modo é possível ampliar ainda mais a biblioteca do *software* permitindo que mais pessoas tenham acesso a rotinas que antes não estavam programadas. A Fig. 3 e 4 apresentam o funcionamento do bloco nativo para Reed-Solomon. Como pode ser observado em na Fig. 3 uma fonte randômica está gerando 1000 amostras de '0s' e '1s' que são codificadas e decodificadas pelos blocos RS antes de serem medidas e comparadas com o *bitstream* original. Como visto na Fig.4. as formas de onda se sobrepõem após a decodificação.

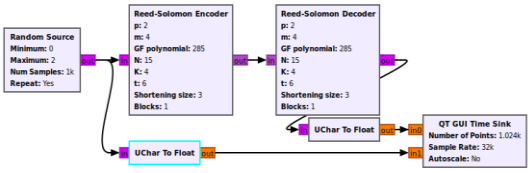

Fig.3. Fluxograma RS – *Encode* e *Decode*

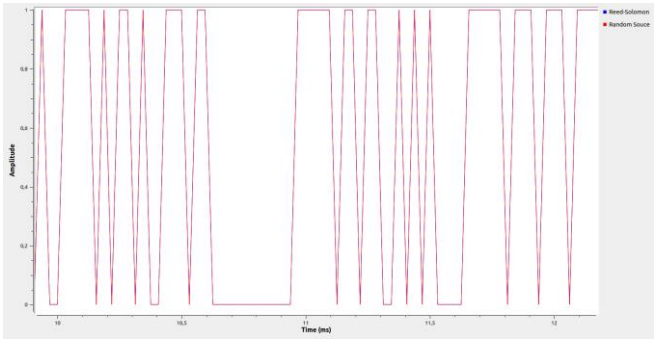

Fig.4. Comparação entre *bitstream* original e *bitstream* pós processamento RS

Os blocos de RS foram estruturados seguindo o padrão [2] com o Galois field [GF(16)] (formado por ‘p^m’), o polinômio primo 0x11d, e na configuração RS(15,4). O *padding* é definido por ‘s’. Para o *interleaver* utilizarou-se os blocos *Convolutional Interleaver/Deinterleaver*, os quais têm por incumbência tornar um bloco de ‘n’ bytes na entrada em uma sequência de um byte na saída. Na Fig 5 e 6 é evidenciado o funcionamento do *puncturing*. Utilizando-se do vetor [1,0,0,1,0,1,1,1] como fonte de sinal, o fluxo de blocos mostrado na Fig.5 foi capaz de remover e preencher de maneira correta os bits referentes ao decimal 252, que em binário apresenta os dois últimos dígitos menos significativos sendo ‘0’.

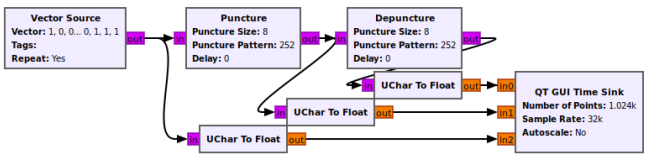

Fig.5. Fluxograma *Puncturing* e *Depuncturing*

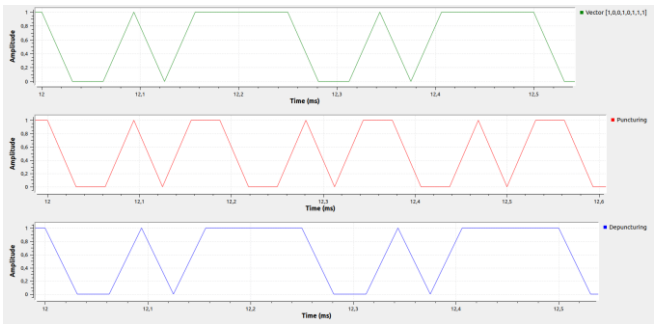

Fig.6. Comparação entre *bitstream* original, *bitstream* pós-*Puncturing* e *bitstream* pós-*Depuncturing*

Observa-se que na forma de onda bem de cima (em verde) o *bitstream* original apresenta dois ciclos enquanto que na forma de onda do meio (em vermelho) o *bitstream* completa aproximadamente três ciclos no mesmo intervalo de tempo. Isto se dá porque após o *puncturing* de 252, os dois últimos bits de cada byte foram cortados, transformando-se assim em símbolos com 6 bits. De modo similar, na forma de onda bem de baixo (em azul) o processo de *depuncturing* faz o preenchimento correto dos bytes com os bits faltantes.

O segundo bloco para correção de erro, o CC, ou também chamado *Inner FEC Encoder/Decoder* não consta na biblioteca de blocos nativos do GNU Radio. Devido à complexidade para implementa-lo ao ambiente do GNU Radio ao envolver o “recorte” de vetores, optou-se por não utilizar este bloco de processamento de sinal.

O *software* GNU Radio não conta com os códigos RLL Manchester nem 4B6B dentro de sua biblioteca, deste modo, utilizando a ferramenta de programação e adição de blocos, foram inseridos estes dois blocos RLL. A Fig. 7 e 8 mostram como o bloco Manchester funciona no GNU Radio enquanto a Fig. 9 e 10 apresentam o bloco 4B6B em operação.

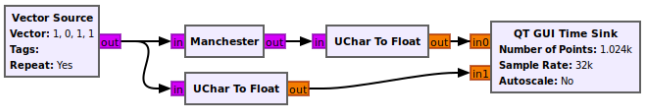

Fig.7. Fluxograma Manchester – *Encode*

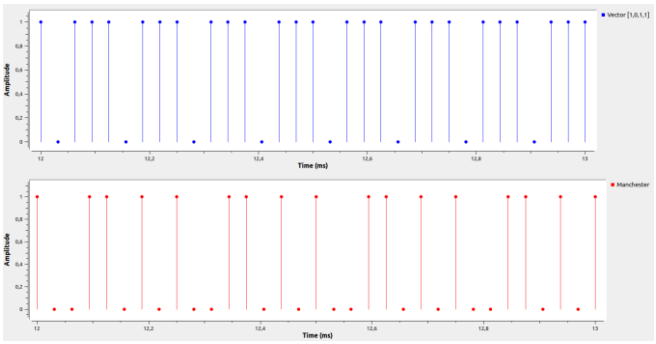

Fig.8. Comparação entre *bitstream* original e *bitstream* pós processamento bloco RLL Manchester

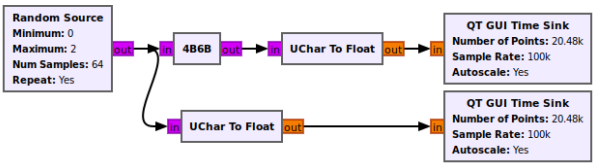

Fig.9. Fluxograma 4B6B – *Encode*

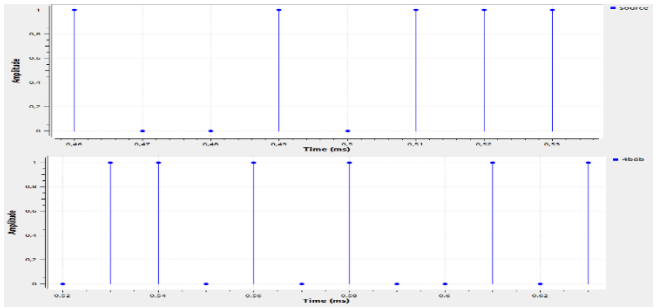

Fig.10. Comparação entre *bitstream* original (em cima) e *bitstream* pós processamento bloco RLL 4B6B (embaixo)

Como pode-se observar no caso Manchester o bit em nível alto ‘1’ se torna o conjunto ‘1’ e ‘0’, nesta ordem, enquanto que o bit ‘0’ se transforma em ‘0’ e ‘1’. Assim, o *bitstream* de [1,0,1,1] se transforma em [1,0,0,1,1,0,1,0] apresentando, portanto, duas vezes mais amostras na saída do bloco, se caracterizando desta vez como um bloco interpolador, dentro da *engine* do GNU Radio. Quanto ao bloco 4B6B verifica-se que o *bitstream* [1,0,0,1,0,1,1,1] se transforma em [0,1,1,0,1,0,1,0,0,1,0,1] aumentando o número de amostras na saída em 1,5 vezes sendo também um bloco do tipo interpolador. Observa-se que a regra utilizada para a conversão do *bitstream* em ambos os casos é segundo [2] para a camada PHY-I.

Por fim, as modulações OOK e VPPM não são nativas do GNU Radio e, portanto, devem ser programadas externamente e adicionadas via gr_modtool dentro da interface do *software*.

Os experimentos realizados foram feitos utilizando um USRP 2944 da National Instruments [11] (Fig. 11) com a transmissão a partir de um LED CREE XLamp XR-C [12] e recepção do sinal através de um fotodetector PDA10A2 fabricado pela Thorlabs [13]. Neste experimento, o transmissor e o receptor foram posicionados em uma bancada de testes e afastados apenas 10 centímetros um do outro, como pode ser visto na Fig. 12. Assim, por fim, a modulação utilizada para realizar os testes de comunicação foi a BPSK que modula o sinal entre o nível '+1' e '-1'. Devido a modulação BPSK ser nativa do GNU Radio ao contrário da OOK que não consta na biblioteca de blocos, sua aplicação se justificou devido a ela ser uma modulação de dois níveis, de modo similar a OOK, diferindo unicamente que para o sinal baixo a OOK envia '0' e a BPSK '-1', deste modo, quando aplicada ao VLC, o sinal transmitido apresenta a mesma mensagem para os níveis altos e baixos. Os blocos para processamento do sinal no GNU Radio podem ser observados na Fig. 13.

Fig.11. Modelo USRP 2944 [11]

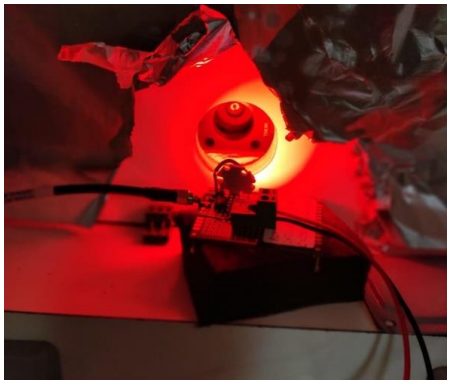

Fig.12. Montagem do Tx e Rx

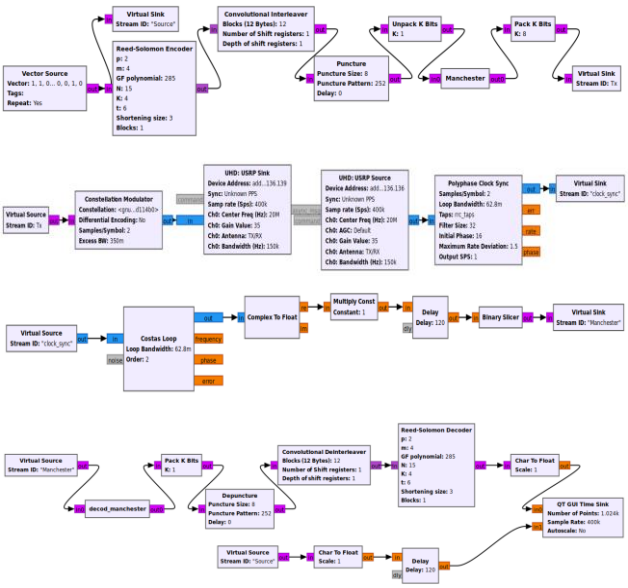

Fig.13. Fluxograma para Modulação BPSK

Começando pelos blocos da parte superior, tem-se a parte de processamento digital do transmissor, contemplando o RS, *Interleaver*, *Puncture* e Manchester. Logo abaixo estão os blocos do USRP, onde ligado ao transmissor está o bloco referente a modulação BPSK e ligado ao receptor está o *Polyphase Clock Sync* responsável por iniciar a sincronização nos sistemas PAM a fim de buscar máximo de SNR para cada amostra, bem como reduzir o *Inter Symbol Interference* (ISI) [8]. Logo em seguida estão os blocos referentes à sincronização do sinal obtido com o sinal enviado em relação ao tempo, neste estágio o uso do *Costas Loop* e o *Delay* fornecem o suporte necessário. Por fim, tem-se os blocos para processamento digital do sinal recebido pelo Rx do USRP, contemplando todos os blocos de decodificação existentes no Tx. Nota-se que o uso do código RLL Manchester invés do 4B6B se deu devido à Standard IEEE 802.15.7 defini-lo como bloco padrão para comunicações por luz visível que utilizem a modulação OOK. O bloco RLL 4B6B apesar de consumir menos dados para a codificação da mensagem, segundo o padrão, deve ser utilizado para a modulação VPPM.

Como resultado final para o experimento, foi realizada a transmissão de um sinal randômico de 64 bits, repetidamente, a fim de verificar a integridade do sinal recebido através da BER (do inglês - *Bit Error Rate*) do sistema. Ao codificar um bloco no GNU Radio para armazenar os dados transmitidos e recebidos em um arquivo e compará-los, torna-se possível verificar a incidência de erros ao longo do tempo. Deste modo foram realizados 4 (quatro) ensaios nos quais a corrente elétrica no Tx (I_{led}) foi diminuída em 0,5A sucessivamente, onde para o primeiro ensaio I_{led1} = 3,5A, no segundo I_{led2} = 3A, no terceiro I_{led3} = 2,5A e no último ensaio I_{led4} = 2A. Observa-se que para todos os ensaios a distância entre Tx e Rx manteve-se inalterada, bem como a tensão no Tx.

Para o cálculo da BER tem-se:

$$BER = \frac{bits\ recebidos\ com\ erro}{bits\ totais\ transmitidos} \quad (1)$$

Primeiramente, foi averiguado se o sinal estava de fato sendo transmitido. A Fig.14 mostra o sinal do primeiro ensaio, observado por um analisador de espectro.

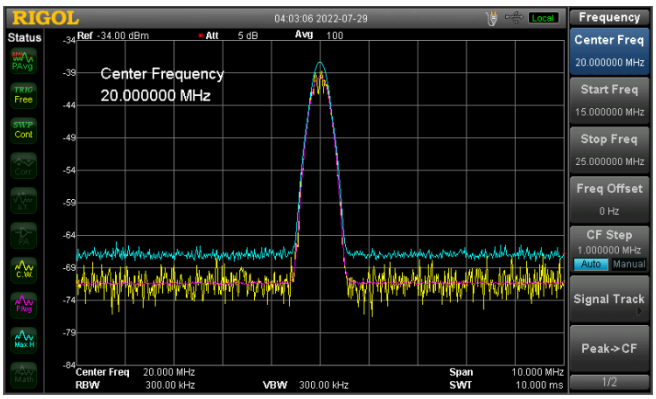

Fig.14. Sinal observado pelo analisador de espectro

Como mostrado na Fig.14, o sinal transmitido, representado pela amplitude próxima a -39dbm, tem sua frequência central em 20MHz e uma largura de banda de 300kHz, valores estes que coincidem com os que foram configurados no próprio GNU Radio para que o USRP operasse.

Utilizando-se dos dados armazenados em bloco de notas com as informações referentes aos bits enviados e recebidos,

e ao compará-los, verificou-se que para o total de 900.000 bits enviados em cada ensaio, em todos os casos o número de bits que foram recebidos com erro excedeu o milhar. Deste modo, aplicando (1) para cada ensaio realizado, verificou-se que a BER para todos eles foram de 10^{-3}, com um pequeno aumento no número de bits recebidos com erro cada vez que se diminuía a corrente no LED, como pode ser observado na Fig.15.

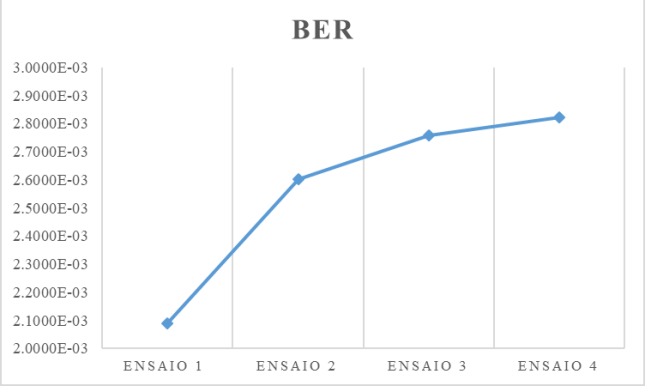

Fig.15. BER do sistema para cada ensaio

V. CONCLUSÕES

Assim sendo, este trabalho apresentou o ambiente de trabalho do GNU para controle do SDR e introduziu e demonstrou o funcionamento de alguns dos blocos necessários definidos pelo *standard* IEEE 802.15.7-2018, como o corretor de erro *Reed-Solomon* com *padding* embutido, *puncturing*, *Manchester* e 4B6B, além da modulação BPSK para envio e recebimento de sinal através de radiofrequência. Percebe-se que há muitas possibilidades para se trabalhar visando otimizar o funcionamento do sistema e obter resultados mais satisfatórios.

A partir da implementação de novos blocos de programação para as modulações é possível tornar o ambiente do GNU Radio ainda mais completo e preciso para o seu uso em VLC. Portanto, este trabalho buscou provar a utilização e valor do *software* para o controle da transmissão e recepção dos dados conforme os resultados apresentados.

Nota-se que ao desenvolver os blocos RLL utilizou-se a programação em Python para escrever as linhas de códigos, e por serem programáveis o seu modo de operação e otimização mudará de programador para programador. Portanto, salienta-se a necessidade de se levar em consideração a otimização do sistema para que este não cause problemas ao processar os dados, evitando assim erros causados por interferência do computador.

Por fim, tem-se que para o sistema proposto, ao variar a corrente no LED transmissor, verificou-se o aumento no número de bits errados recebidos na mensagem. Observa-se que neste experimento, a não utilização do Convolutional Code para a correção de erros, como é requerido no padrão IEEE 802.15.7-2018, certamente foi um dos fatores determinantes para a BER obtida nos ensaios. Além disto, verificou-se que a variação na potência do LED através do controle da corrente elétrica também foi um fator relacionado ao aumento de erro na comunicação.

VI. REFERÊNCIAS

[1] U. S. Department of Energy, “Energy Savings Forecast of Solid-State Lighting in General Illumination Applications,” 2019.

[2] IEEE Standards Association, “IEEE Std 802.15.7 - 2018,” IEEE Computer Society, New York, 2018.

[3] W. Hussain, H. F. Ugurdag, M. Uysal, “Software Defined VLC System: Implementation and Performance Evaluation,” em 2015 4th International Workshop on Optical Wireless Communications (IWOW), Istanbul, 2015.

[4] J. Baranda, P. Henarejos, C. G. Gavrincea, “An SDR implementation of a visible light communication system based on the IEEE 802.15.7 standard,” em International Conference on Telecommunications, Casablanca, 2013.

[5] Ettus Research, “Universal Software Radio Plataform,” 2013.

[6] Y. Qiao, H. Haas, E. Knightly, “A Software-defined Visible Light Communications System with WARP,” em 1st ACM MobiCom Workshop on VLC Systems, 2014.

[7] WARP, “Warp Project”.

[8] GNU Radio, “GNU Radio the Free & Open Software Radio Ecosystem,” 2022.

[9] W. Geisel, “Tutorial on Reed-Solomon Error Correction Coding,” Houston, 1990.

[10] B. Turan, O. Narmanlioglu, S. C. Ergen, M. Uysal, “Physical Layer Implementation of Standard Compliant Vehicular VLC,” em IEEE 84th Vehicular Technology Conference (VTC-Fall), Montreal, 2016.

[11] National Instruments, “USRP 2944”.

[12] CREE LED, “XLamp XR-C - Red,” 2021.

[13] Thorlabs, “PDA10A2 Si Amplified Fixed Gain Detector,” 2019.